Specification

1. Title of the invention

INFORMATION PROCESSING SYSTEM AND METHOD

5   2.   Background of the invention

The present invention relates to an information processing apparatus, a method for controlling the information processing apparatus, a computer program to implement the functions of the above apparatus, and an information processing system. More particularly, the invention relates to the information processing apparatus that makes effective use of a cache memory when reading data from storage equipment.

Data amount to be handled by an information processing system has lately been growing. Correspondingly, the capacity of storage equipment for use in such a system has become larger and the storage equipment is required to have higher input/output performance and reliability commensurate with its larger capacity. In view hereof, storage network technology which uses multiplex logical paths (input/output paths) to storage equipment and appropriately allocates data input/output requests (I/O) requests to the logical paths has been developed (for example, W. Curtis Preston, "Using SANs and NAS, Help for Storage Administrators" O'Reilly, Feb. 2002, pp. 54-55).

In the foregoing prior art, the storage equipment executes processes in response to I/O requests in order in

which it received the I/O requests from its host information processing apparatus.

However, the I/O requests that the storage equipment receives include conjunct ones and a mechanism for
5 efficiently executing the processes in response to the I/O requests, focusing attention on the distinctiveness of the I/O requests, has not been developed. In particular, when a load balancing function is performed, I/O requests are evenly distributed across the paths and I/O requests for
10 access to sequential locations of data, namely, sequential access, also may be allocated to different paths. The storage system side cannot distinguish I/O requests for sequential data access and, consequently, it has occurred that prefetch caching does not function effectively, which
15 results in a decrease in access performance.

A method of allocating sequential access I/O requests to a same path to process them has been proposed. However, if a plurality of applications get access to a same logical unit, I/O requests from the applications are serialized and,
20 therefore, sequential access I/O requests from one application are not always to be sequential in access to successive locations on the logical unit. In this case, it also occurs that prefetch caching does not function effectively, which results in a decrease in access
25 performance.


3. Summary of the invention

An object of the present invention is to provide an information processing system that enables efficient execution of I/O processes through effective use of prefetch caching by allocating sequential I/O requests to a same path.

5      A primary aspect of the present invention may be defined as follows. In an information processing system comprising a storage equipment which includes a logical unit logically assigned to physical devices and an information processing apparatus which sends data input/output requests

10      to the storage equipment, wherein the data input/output requests are transferred through logical paths serving as communication paths to the logical unit, a cache memory is provided to prefetch and store both data in a location to be accessed by an input/output request and data in locations

15      following that location within the physical devices and the information processing apparatus comprises a path selection management section which manages configurations of a plurality of blocks into which the logical unit is divided, an I/O request allocation section which allocates data

20      input/output requests to be transmitted to the storage equipment to the logical paths, and I/O processing units which transmit the data input/output requests through the logical paths, according to the allocation determined by the I/O request allocation section, pursuant to an established

25      protocol. The path selection management section assigns at least one logical path to one block.

In the present invention, because the path selection

management section which manages configurations of a plurality of blocks into which a logical unit is divided assigns at least one logical path to one block, prefetch caching can be used effectively.

5

4.  Brief description of the drawings

FIG. 1 is a block diagram showing an overall structure of an information processing system according to a preferred embodiment of the present invention;

10      FIG. 2 is a block diagram of the information processing system of the embodiment of the present invention, including a detailed configuration of a path management unit;

FIG. 3 is a functional block diagram to explain creating tables in an even blocking mode, the tables being

15      used in the embodiment of the present invention;

FIG. 4 shows an illustrative path management table that is used in the embodiment of the present invention;

FIG. 5 shows an illustrative LU management table that is used in the embodiment of the present invention;

20      FIG. 6 shows an illustrative block management table that is used in the embodiment of the present invention;

FIG. 7 is a functional block diagram to explain creating the tables in a variable blocking mode, the tables being used in the embodiment of the present invention;

25      FIG. 8 is another functional block diagram to explain creating the tables in the variable blocking mode, the tables being used in the embodiment of the present invention;

FIG. 9 shows an illustrative blocks assignment table that is used in the embodiment of the present invention;

FIG. 10 is a flowchart of a procedure for creating the path management table, according to an embodiment of the
5  present invention;

FIG. 11 is a flowchart of a procedure for updating the path management table, according to an embodiment of the present invention;

FIG. 12 is a flowchart of a procedure for updating the
10  LU management table, according to an embodiment of the present invention;

FIG. 13 is a flowchart of operation of a block management section, according to an embodiment of the present invention; and

15  FIG. 14 is a flowchart of I/O processing, according to an embodiment of the present invention.

3.  Detailed description of the preferred embodiments

Preferred embodiments of the present invention will
20  be described hereinafter with reference to the accompanying drawings.

FIG. 1 is a block diagram showing an overall structure of an information processing system according to a preferred embodiment of the invention.

25  The information processing system comprises at least one unit of information processing apparatus 100 and at least one unit of storage equipment 200.  In the present embodiment,

the information processing system comprised of one unit of
information processing apparatus 100 and one unit of storage
equipment 200 is discussed.  However, the information
processing system may be comprised of two or more units of
5    information processing apparatus 100 and two or more units
of storage equipment 200.

To the information processing apparatus 100, the
storage equipment 200 is connected via a network (for example,
Storage Area Network (SAN)).  Communication between the
10   information processing apparatus 100 and the storage
equipment 200 via the SAN is generally performed, pursuant
to a Fibre Channel Protocol.  Specifically, from the
information processing apparatus 100, a request to access
data in units of blocks is transmitted to the storage
15   equipment 200, pursuant to the Fiber Channel Protocol.  The
information processing apparatus 100 and the storage
equipment 200 are not necessarily connected via the SAN; they
may be connected via a network such as LAN or may be directly
connected through a Small Computer System Interface (SCSI).
20      A plurality of paths (logical paths) 301 to 304 are
set up to constitute the SAN connecting the information
processing apparatus 100 and the storage equipment 200.  The
information processing apparatus 100 accesses data stored
on the storage equipment 200 through the plurality of logical
25   paths 301 to 304.  The logical paths 301 to 304 are associated
with physical paths that are data transfer lines, physically
provided by hardware, to connect the information processing

apparatus 100 and the storage equipment 200.

The information processing apparatus 100 includes a CPU (Central Processing Unit), a memory, etc. and executes various types of application programs. The information

5    processing apparatus 100 comprises an application 110, a path management unit 120, a plurality of I/O processing units 131 to 134, and a plurality of adapters 141 to 144.

The application 110 represents diverse application programs (for, example, programs for a bank's automatic

10    teller system, a seat reservation system of airlines, etc.) to be run on the information processing apparatus 100. The application program performs tasks, while accessing data stored on the storage equipment 200.

The adapters 141 to 144 are interfaces for

15    communication with the storage equipment 200; for example, SCSI adapters or Host Bus Adapters (HBAs) such as Fibre Channel adapters. Through the adapters 141 to 144, the information processing apparatus 100 can send data input/output requests (I/O requests) to and receive data

20    from the storage equipment 200.

The I/O processing units 131 to 134 are realized by drivers and software for controlling the adapters 141 to 144 (for example, SCSI drivers). The I/O processing units 131 to 134 processes I/O requests delivered from the path

25    management unit 120 sequentially. The I/O processing units 131 to 134 convert I/O requests into packets of an established communication protocol (Fibre Channel, TCP/IP, etc.) and

transmit the packets through the adapters 141 to 144 to the storage equipment 200. In the embodiment as will be set forth herein below, the storage equipment 200 is provided with a data buffer 211. However, the I/O processing units

5   131 to 134 or the adapters 141 to 144 may be provided with the data buffer.

Path management software runs on the path management unit 120 to manage the logical paths 301 to 304 between the information processing apparatus 100 and the storage

10  equipment 200. The path management unit 120 multiplexes the logical paths 301 to 304 through which I/O requests are transmitted to the storage equipment 200 and expands the bandwidth of each path, thereby preventing a bottleneck in data access and enhancing data I/O processing performance.

15  The path management unit 120 allocates I/O requests to the logical paths 301 to 304 for load balancing. Moreover, the path management unit 120 has a function to disconnect a logical path, one of the paths 301 to 304, in which a fault has occurred and allow access to data on the storage equipment

20  200 to continue through the remaining sound ones among the logical paths 301 to 304. This function avoids service stoppage due to a logical path fault and boosts the reliability of the system.

While the information processing apparatus 100 is

25  assumed to be a computer (personal computer, work station, mainframe computer, etc.) in the description of the present embodiment, the information processing apparatus 100 may be

a network device such as a router and a hub.

Access from the information processing apparatus 100 to data stored on the storage equipment 200 is performed by an I/O request such as, for example, a data write request or a data read request. This I/O request is issued from the application 110 that is one of a variety of applications. In the I/O request, a header, storage equipment, logical unit, address, data length, etc. are specified. The header contains the identifier of the information processing apparatus 100 that sends the I/O request. The storage equipment field contains the identifier of the storage equipment to which the I/O request is to be delivered and processed. The logical unit field contains the identifier of a logical unit from which data is to be read or to which data is to be written by the I/O request in the storage equipment. The address field contains the address (start address) at which data read or write will start. The data length field contains the size (length) of the data to read or write. If the I/O request is a data write request, the data to write is added to the request.

I/O requests issued from the application are distributed among the logical paths 301 to 304 by the path management unit 120 (for example, the requests are evenly allocated among the paths) and transmitted to the storage equipment 200. By parallel transfer of a plurality of I/O requests in this way, the processing performance of the information processing system can be enhanced.

The storage equipment 200 comprises a disk control unit 210, a logical unit (LU) 220, and other entities and performs data input/output processing, according to the I/O request received from the information processing apparatus 100.

5      The logical unit 220 is a logically set up unit of storage space from storage resources (physical devices) which are provided to the information processing apparatus 100. In FIG. 1, the storage equipment 200 has one logical unit; however, may have two of more logical units. The

10     logical unit 220 is divided into a plurality of blocks 221 to 224 and the logical paths 301 to 304 are set linked to the blocks 221 to 224 in one-to-one correspondence. As the storage resources, a variety of storage media such as hard disk devices (disk array devices), flexible disk devices,

15     and semiconductor storage devices can be used.

The disk control unit 210 receives I/O requests transmitted from the information processing apparatus 100 and controls access to data stored on the logical unit 220. The disk control unit 210 has a data buffer 211. The data

20     buffer 211 is provided to be used commonly for the logical paths 301 to 304. Data buffers may be provided for each of the logical paths 301 to 304. The data buffer 211 functions as a cache memory into which data to write to the logical unit 220 or data read from the logical unit 220 is stored

25     temporarily.

In the present embodiment, the storage equipment 200 is provided with a prefetch caching function. The prefetch

caching function works as follows. If a location on the logical unit 220 specified in an I/O request that has just been received through a logical path by the storage equipment 200 succeeds a location on the logical unit 220 specified

5    in an I/O request previously received through the same logical path, the prefetch caching function prefetches the data in the location (sequential locations) following the location on the logical unit 220 specified in the previous I/O request from the logical unit 220 and stores that data

10   into the data buffer 211 in advance. This prefetch caching function enables effective use of data existing in the data buffer 211 when I/O requests that specify consecutive locations on the logical unit 220 are received through the same logical path from the information processing apparatus

15   100 and, consequently, time required for data input/output processing can be shortened.

When data write requests are transmitted from the information processing apparatus 100 through the logical paths 301 to 304, an action such as read, modify, or write

20   is performed on the data buffer 211 in the storage equipment 200.

For example, if a data write request is transmitted from the information processing apparatus 100 through a logical path, one of 301 to 304, the storage equipment 200

25   writes the data into the data buffer 211. Upon the completion of writing the data into the data buffer 211, the storage equipment 200 transmits a data write request

complete notification (I/O complete notification) back to the information processing apparatus 100 through the same logical path, one of 301 to 304, through which the data write request (I/O request) was transmitted, regardless of whether

5   or not writing of the data to the logical unit 220 is complete. In other words, the I/O complete notification is sent to the information processing apparatus 100, not synchronously with actual data writing to the logical unit 220. After that, the disk control unit 210 writes the data that has been

10  written into the data buffer 211 to the logical unit 220.

If a data read request is transmitted from the information processing apparatus 100 through a logical path, one of 301 to 304, the storage equipment 200 judges whether the data to read exists in the data buffer 211. If the data

15  to read exists in the data buffer 211, the storage equipment 200 reads the data from the data buffer 211 and transfers the data that has existed in the data buffer 221 to the information processing apparatus 100 without reading the data from a disk.

20  Otherwise, if the data to read does not exist in the data buffer 211, the disk control unit 210 reads the data in the location specified in the data read request from the logical unit 220 and transfers that data to the information processing apparatus 100.

25  At this time, when the disk control unit 210 reads the data from the location specified in the received data read request on the logical unit 220, if locations (sequential

locations) following the specified location exist, the disk control unit 210 reads data in the subsequent sequential locations (sequential data) from the logical unit 220 and stores the thus read data (sequential data from the specified

5  location and subsequent locations) into the data buffer 211.

By the above-described prefetch caching function, data existing in the data buffer 211 can be used and I/O processing time can be shortened, because I/O requests for consecutive locations in a same block on the logical unit

10  220 from the information processing apparatus 100 and responses are transmitted and received through the same logical path.

The disk control unit 210 can be integrated into the storage equipment 200 as shown in FIG. 1 or can be provided

15  as a separate entity.  The logical unit can be divided into a plurality of partitions.

FIG. 2 is a block diagram of the information processing system shown in FIG. 1, including a detailed configuration of the path management unit 120.

20  The path management unit 120 is comprised of an I/O request receiving section 121, an I/O complete notification section 122, an I/O request allocation section 123, a path selection management section 124, a section to judge request sequential 125, a section to create/deliver blocks setup 127,

25  and a section to display access information 128.

The I/O request receiving section 121 receives I/O requests issued from the application 110.  The I/O complete

notification section 122 notifies the application 110 of the completion of processing of an I/O request that the I/O request receiving section 121 received from the application 110.

5      The section to judge request sequential 125 judges whether an I/O request that the I/O request receiving section 121 has just received from the application 110 is sequential I/O request.  Specifically, the section to judge request sequential 125 judges whether a storage location specified
10  in the just received I/O request (that is assumed to be the second I/O request of sequential I/O access) succeeds a storage location specified in an I/O request (that is the first one of the sequential I/O) received from the application 110 before the reception of the second I/O
15  request.  This judgment is made by comparing the just received request (second) to all I/O requests previously stored in the data buffer 211 as to whether:

(1) the storage equipment specified in the second I/O request matches the storage equipment specified in the first I/O
20  request;

(2) the logical unit specified in the second I/O request matches the logical unit specified in the first I/O request; and

(3) the address specified in the second I/O request is equal
25  to the address plus the data length specified in the first I/O request.

If the first I/O request that meets all the foregoing

judgment criteria (1) to (3) exists, the section to judge request sequential 125 judges the second I/O request to be sequential I/O request. Otherwise, if there is not the first I/O request that meets the foregoing judgment criteria (1)

5     to (3), the section to judge request sequential 125 judges the second I/O request to be random I/O request. This judgment as to whether the just received I/O request is sequential is made by comparing the second I/O request of data read to previously buffered I/O requests of data read

10    to find the first one of sequential read and comparing the second I/O request of data write to previously buffered I/O requests of data write to find the first one of sequential write.

       After judged sequential or random by the section to

15    judge request sequential 125, the I/O requests are queued into an I/O request queue 126 until they are allocated.

       The I/O request allocation section 123 determines one of the I/O processing units 131 to 134 to which to allocate an I/O request from the I/O request queue 126 and selects

20    one of the logical paths 301 to 304 to be used to transmit the I/O request. The I/O processing units are provided for the paths, respectively, and process the I/O requests allocated to the paths and get access to the storage equipment 200.

25       When an I/O request is allocated by the I/O request allocation section 123, a path assigned to a block where data to be acted upon by the I/O request exists (block including

the address specified in the I/O request) is selected.  The

I/O request allocation section 123 allocates I/O requests

to distribute them evenly among the paths by the load

balancing function.  For example, the I/O request allocation

5     section 123 preferentially allocates an I/O request to an

I/O processing unit, one of 131 to 134, that is now idle or

an I/O processing unit, one of 131 to 134, that has completed

I/O request processing.

The I/O request allocation section 123 includes a

10    section to detect path fault 123a and a section to detect

path recovery 123b which realize a path status check function

to detect change in the statuses of the paths to the storage

equipment 200.  Specifically, when user operation has been

performed to place a path in an offline state (a blocked state

15    that the path cannot transmit an I/O request normally due

to a fault occurring in the path) or upon the detection of

a path fault during I/O request processing, the section to

detect path fault 123a regards the path status as having

changed and notifies a path management section 124a of the

20    path fault.  When the path recovers by an automatic path

recovery function that each of the I/O processing units 131

to 134 has, the section to detect path recovery 123b detects

the recovery of the path from the fault and notifies the path

management section 124a of the path recovery.  This

25    automatic path recovery function attempts transmission of

an I/O simulated signal through the path that is inactive

at predetermined timing, for example, at certain intervals,

and checks whether the signal has been transmitted normally, thereby checking whether the path has recovered, not by explicit user operation to return the inactive path to the active state.

5      The path selection management section 124 is comprised of the path management section 124a, a blocks setup management section 124b, and a block management section 124c. The path management section 124a creates a path management table 2100 (FIG. 4). The blocks setup management section 10     124b creates an LU management table 2200 (FIG. 5), referring to the path management table 2100. The block management section 124c creates a block management table 2300 (FIG. 6), referring to the LU management table 2200.

The section to create/deliver blocks setup 127 is 15     provided to be used in a variable blocking mode which will be described later. The section to create/deliver blocks setup 127 creates a blocks assignment table 2400, based on operator input, and retains the table.

The section to display access information 128 outputs 20     sequential access I/O counts and random access I/O counts per block. For example, this section outputs information (such as sequential/random access counts) necessary for the user to change settings. In this case, the user must be conscious of dividing the logical unit into blocks and change 25     settings.

FIG. 3 is a functional block diagram to explain creating the tables that are used in the embodiment of the

present invention.

The path management section 124a creates the path management table 2100 at the start of the path management software. The path management table 2100 contains as many path entries as the number of paths that are recognizable at the start of the path management software.

The path management section 124a also updates the sequential I/O count 2104 or the random I/O count 2105, based on the result of judgment made by the section to judge request sequential 125 as to whether a normally processed I/O request is sequential access or random access.

Moreover, the path management section 124a updates the path status 2103 in the path management table 2100 when it receives a notification of path status change detected from the section to detect path fault 123a or the section to detect path recovery 123b. Upon the completion of updating the path management table 2100, the path management section 124a sends an LU data update request to update the LU management table 220 to the blocks setup management section 124b. At this time, the path management section 124a notifies the blocks setup management section 124b of the ID of the path that has become faulty or recovered.

In the variable blocking mode which will be described later, the path management section 124a, upon the completion of creating or updating the path management table 2100, sends a blocks setup request to create the blocks assignment table 2400 to the section to create/deliver blocks setup 127.

When receiving the request to create the LU management table 2200 from the path management section 124a, the blocks setup management section 124b creates the LU management table 2200, referring to the path management table 2100

5   created by the path management section 124a.

When receiving the request to update the LU management table 2200 from the path management section 124a, the blocks setup management section 124b updates the LU management table 2200, based on the ID information of the path that has

10  become faulty or recovered, notified with the LU management table update request.

Specifically, in the case of a path fault notification from the section to detect path fault 123a, the blocks setup management section 124b refers to the LU management table

15  2200, extracts the path that was known to be faulty by the notification, identifies a logical unit to which the path connects, and decrements the number of paths 2203 for the logical unit by one in the LU management table 2200. Then, the blocks setup management section 124b deletes the ID of

20  the faulty path notified with the LU management table update request from the path IDs 2204 field for the logical unit.

In the case of a path recovery notification from the section to detect path recovery 123b, the blocks setup management section 124b refers to the path management table

25  2100, identifies a logical unit to which the path that was known to have recovered by the notification connects, and increments the number of paths 2203 for the logical unit by

one in the LU management table 2200. Then, the blocks setup management section 124b adds the ID of the recovered path notified with the LU management table update request to the path IDs 2204 field for the logical unit.

5        Upon the completion of creating the LU management table 2200, the blocks setup management section 124b requests the block management section 124c to create the block management table 2300. Upon the completion of updating the LU management table 2200, the blocks setup management section

10    124b requests the block management section 124c to update the block management table 2300.

When receiving the request to create or update the block management table 2300 from the blocks setup management section 124b, the block management section 124c creates or

15    updates the block management table 2300, referring to the LU management table 2200 created by the blocks setup management section 124b. For example, at the start of the path management software, a request to create the block management table is issued and the block management table

20    2300 is created.

Moreover, the block management section 124c notifies the I/O request allocation section 123 of the IDs of the paths that are used to access the blocks.

FIG. 4 shows the path management table that is used

25    in the embodiment of the present invention.

The path management table 2100 maintains updated information about the paths that are recognizable at the

start of the path management software. That is, a new path management table 2100 is created at the start of the path management software and its information is updated when a path fault occurs or when the faulty path has recovered.

5      The path management table 2100 is created in both even blocking mode and variable blocking mode. The even blocking mode is to divide a logical unit by the number of active paths into a plurality of blocks of even capacity (size) and assign one path to one block. On the other hand, the variable

10     blocking mode is to divide a logical unit into a plurality of blocks, according to the blocks assignment table 2400 (FIG. 7).

The path management table 2100 is created by the path management section 124a and the blocks setup management

15     section 124b references this table when creating (or updating) the LU management table 2200. The path management table 2100 is always maintained during the run of the path management software.

The path management table 2100 contains the following

20     information items: path ID 2101, logical unit (LU) 2102, path status 2103, sequential I/O count 2104, and random I/O count 2105. These information items per path ID are stored.

The path ID 2101 is the identifier assigned to a path by the path management unit 120 and path IDs are associated

25     with logical paths in one-to-one correspondence. The logical unit (LU) 2102 is the identifier assigned to a logical unit in the storage equipment and one or a plurality of paths

are set up to terminate at one logical unit. One or a plurality of path IDs is associated with one logical unit.

The path status 2103 indicates whether or not the path identified by the path ID 2101 entry is capable of operating.

5    That is, if the path is in the active state that it can transmit an I/O request normally, the path status field contains "online." Otherwise, if a fault occurs in the path and the path is placed in the blocked state that it cannot transmit an I/O request normally, the path status field

10   contains "offline."

The sequential I/O count 2104 field contains a total count of sequential access I/O requests transmitted through the path as the result of the judgment as to whether an I/O request is sequential access or random access, made by the

15   section to judge request sequential 125. The random I/O count 2105 field contains a total count of random access I/O requests transmitted through the path as the result of the above judgment. The sequential IO count 2104 and the random I/O count 2105 are initialized to "0."

20   FIG. 5 shows the LU management table that is used in the embodiment of the present invention.

The LU management table 2200 maintains the configuration states of logical units. The blocks setup management section 124b creates (or updates) this table,

25   referring to the path management table 2100. After the path management table 2100 is created (or updated), the LU management table 2200 is referenced when the block

management section 124c creates (or updates) the block
management table 2300. The LU management table 2200 is
created in both even blocking mode and variable blocking
mode.

5        The LU management table 2200 contains the following
information items: logical unit (LU) 2201, storage capacity
(LU size) 2202 of the logical unit, the number of paths 2203
that are placed in the active state, and path IDs 2204 of
the active paths. These information items per logical unit
10   are stored.

The logical unit (LU) 2201 field contains the
identifier of each of the logical units 2102 entered in the
path management table 2100, wherein duplicated information
about a same LU is eliminated. The field of storage capacity
15   (LU size) 2202 of the logical unit contains the storage
capacity of the logical unit entered in the logical unit 2201
field.

The number of paths 2203 field contains the number of
active paths connected to the logical unit entered in the
20   logical unit 2201 field. The number of paths is obtained
by summing up the number of paths whose status 2203 is active
(online), associated with the logical unit, in the path
management table 2100. The number of active paths entered
in the number of paths 2203 field is decremented by a path
25   fault occurrence and the number of paths is updated in the
LU management table 2200.

The path IDs 2204 field lists the IDs of all active

paths connected to the logical unit entered in the logical unit 2201 field. In the variable blocking mode, however, the path IDs 2204 need not be listed in the LU management table 2200.

5        FIG. 6 shows the block management table that is used in the embodiment of the present invention.

The block management table 2300 provides information for selecting a favorable path. The I/O request allocation section 123 references this table, when allocating I/O

10    requests delivered from the application 110 to the paths assigned to the blocks, using logical unit and start address information extracted from the I/O requests. The block management section 124c creates the block management table 2300, referring to the LU management table 2200.

15       The block management table 2300 contains the following information items: block ID 2301, block size 2302, logical unit (LU) 2303, start address 2304, path ID 2305, sequential I/O count 2306, and random I/O count 2307. These information items per block ID are stored.

20       The block ID 2301 field contains the identifier assigned to a block by the path management unit 120 and block IDs are associated with blocks in one-to-one correspondence.

The block size 2302 field contains the length of the block entered in the block ID 2301 field. The block

25    management section 124c calculates the block size, referring to the LU management table 2200.

The logical unit (LU) 2303 field contains the

identifier assigned to a logical unit and this is the identifier of the logical unit 2102 associated with the path ID 2203 entered in the path management table 2100.

The start address 2304 field contains the start address of each block obtained by the block management section 124c.

The path ID 2305 field contains the ID of the path that is used to access the block. The path IDs are notified from the path management section 124a and the IDs of only the paths that can be used at the present are entered. If one path is assigned to one block, there is one path ID 2305 field for the block. If a plurality of paths is assigned to one block, there is a plurality of ID path 2305 fields for the block.

The sequential I/O count 2306 field contains a total count of sequential access I/O requests processed in the block (transmitted through the path) as the result of the judgment as to whether an I/O request is sequential access or random access, made by the section to judge request sequential 125. The random I/O count 2307 field contains a total count of random access I/O requests processed in the block (transmitted through the path) as the result of the above judgment. The sequential IO count 2306 and the random I/O count 2307 are initialized to "0."

In the above-mentioned even blocking mode, a logical unit is divided by the number of active paths into a plurality of blocks of even capacity (size). Therefore, when a path fault occurs or when the faulty path has recovered, the block

management table is updated in the same procedure as
performed at the start of the system. When a path fault
occurs, the number of blocks decrements and the block size
increases accordingly. When the faulty path has recovered,
5   the number of blocks increments and the block size decreases
accordingly (in other words, the blocks return to their setup
before the path fault occurs).

FIG. 7 is a functional block diagram to explain
creating the tables in another mode, the tables being used
10  in the embodiment of the present invention.

FIG. 7 represents a process where the tables are
created when the variable blocking mode is applied in which
a logical unit is divided into a plurality of blocks,
according to a rule defined in the blocks assignment table
15  2400, unlike the even blocking mode in which a logical unit
is evenly divided into blocks, which is applied when the
logical unit 220 is divided into blocks, as shown in FIG.
3. Therefore, the section to create/deliver blocks setup
127 retains the blocks assignment table 2400 containing
20  information about the blocks into which a logical unit must
be divided. In the even blocking mode, the section to
create/deliver blocks setup 127 and the blocks assignment
table 2400 are not used.

In the variable blocking mode, although the
25  information processing system starts its operation in the
even blocking mode at the start of the path management
software, the operator specifies blocks setup, based on the

sequential and random access I/O counts output by the section
to display access information 128. When the section to
create/deliver blocks setup 127 creates the blocks
assignment table 2400, according to the specified blocks

5    setup, the information processing system switches to
operation in the variable blocking mode. When a path fault
occurs or when the faulty path has recovered, the blocks
assignment table 2400 is referenced, the size of each block
which is determined, depending on the number of active paths

10   at the present, and the number of paths assigned to each block
are extracted, and the block management table 2300 is
created.

       It may also preferable that the section to
create/deliver blocks setup 127 automatically creates the

15   blocks assignment table 2400 by analyzing the sequential and
random access I/O counts and optimizes the blocks assignment
table 2400 without prompting the operator to specify blocks
setup.

       The path management section 124a creates (or updates)

20   the path management table 2100 and requests the blocks setup
management section 124b to create (or update) the LU
management table 2200, as is the case for FIG. 3.

       The blocks setup management section 124b creates or
updates the LU management table 2200, referring to the path

25   management table 2100, and requests the block management
section 124c to create or update the block management table
2300, as is the case for FIG. 3.

When receiving the request to create or update the block management table from the blocks setup management section 124b, the block management section 124c creates or updates the block management table 2300, referring to the

5    LU management table 2200 created by the blocks setup management section 124b and the blocks assignment table 2400 created by the section to create/deliver blocks setup 127.

FIG. 8 is another functional block diagram to explain creating the tables in the variable blocking mode, shown in

10   FIG. 7, the tables being used in the embodiment of the present invention.

When receiving a request to create blocks setup data, the section to create/deliver blocks setup 127 creates the blocks assignment table 2400.  When receiving a request to

15   update blocks setup, the section to create/deliver blocks setup 127 updates the blocks assignment table 2400.

When receiving a request to create or update the block management table from the section to create/deliver blocks setup 127, the block management section 124c creates or

20   updates the block management table 2300, referring to the LU management table 2200 created by the blocks setup management section 124b and the blocks assignment table 2400 created by the section to create/deliver blocks setup 127.

FIG. 9 shows the blocks assignment table that is used

25   in the variable blocking mode (FIGS. 7 and 8) and in the embodiment of the present invention.

The section to create/deliver blocks setup 127 creates

the blocks assignment table 2400 automatically (or based on user input), based on information presented by the section to display access information and taking account of the frequency and type of access to a logical unit (sequential

5    access or random access). If the blocks assignment table 2400 exists, the block management section 124c creates the block management table 2300, referring to the blocks assignment table 2400. The blocks assignment table 2400 contains information as to how an available logical unit is

10   divided into blocks and what active paths assignments to the blocks.

The blocks assignment table 2400 contains the following information items: logical unit (LU) 2401, the number of active paths 2402 that can be used for the logical

15   unit, start address 2403, and the number of assigned paths 2404 set per block. These information items per logical unit are stored.

The logical unit (LU) 2401 field contains the identifier of a logical unit to which the variable blocking

20   mode is applied among the logical unit 2102 entries in the path management table 2100.

The number of active paths 2402 field contains the number of active paths that can be used for each logical unit and this is used as a blocks setup data search key when a

25   way of dividing the logical unit into blocks is determined.

The start address 2403 field contains the start address of each of the blocks into which the logical unit is divided,

depending on the specified number of paths.

The number of assigned paths 2404 field contains the number of paths assigned to each of the blocks into when the logical unit is divided.

5    It may also be preferable to use an I/O monitor, collect I/O data by the I/O monitor, automatically select an optimum way of dividing a logical unit into blocks, and determine the number of paths.  For example, for blocks to which random access occurs more frequently, it is also possible to apply

10   a normal load balancing method which allocates I/O requests to the paths to spread loads evenly across the paths without applying the path assignments according to the present invention.

The blocks assignment table may be defined to divide

15   a logical unit into blocks corresponding to partitions determined by the OS (operating system).

FIG. 10 is a flowchart of a procedure for creating the path management table, according to an embodiment of the present invention.

20   At the start of the path management software, this procedure is performed by the path management section 124a.

First, the path management section 124a gets the number of paths that are recognized by the I/O request allocation section 123 and assigns unique path IDs to the paths (S1001).

25   For example, path IDs can be assigned to the paths from "0" by order of path information acquisition.  Then, the path management section 124a gets the IDs of logical units to which

the paths connect from the I/O request allocation section 123 (S1002). There is n-to-1 correspondence between path IDs and logical unit IDs.

The path management section 124a enters the thus obtained path IDs and logical unit IDs into the path ID 2102 fields and the logical unit (LU) 2102 fields, respectively, in the path management table 2100 (S1003). Then, the path management section 124a initializes the sequential I/O count 2104 and the random I/O count 2105 to "0" (S1004) and initializes the path status 2103 "online" for all the paths (S1005) in the path management table. Then, the path management section 124a issues a request to create the LU management table 2200 to the blocks setup management section 124b (S1006).

Accordingly, at the start of the path management software, the path management section 124a gets information for all paths that are recognized by the host side 100 and for the logical units that are accessed through the paths, creates the path management table 2100, and requests the blocks setup management section 124b to create the LU management table 2200.

FIG. 11 is a flowchart of a procedure for updating the path management table, according to an embodiment of the present invention, and this procedure is performed by the path management section 124a.

The path management section 124a watches for notification from the section to detect a path fault 123a

and the section to detect path recovery 123b. When receiving a path fault notification from the section to detect a path fault 123a (S1101), the path management section 124a updates the status 2103 of the path "offline" in the path management table 2100 (S1102). Then, the path management section 124a issues a request to update the LU management table 2200 to the blocks setup management section 124b (S1103). The path management section 124a sends the ID of the faulty path with the LU management table update request.

When receiving a path recovery notification from the section to detect path recovery 123b (S1104), the path management section 124a updates the status 2103 of the path "online" in the path management table 2100 (S1105). Then, the path management section 124a issues a request to update the LU management table 2200 to the blocks setup management section 124b (S1106). The path management section 124a sends the ID of the recovered path with the LU management table update request.

FIG. 12 is a flowchart of a procedure for updating the LU management table, according to an embodiment of the present invention, and this procedure is performed by the blocks setup management section 124b.

When receiving the LU management table update request issued by the path management section 124a, the blocks setup management section 124b determines whether the update request is due to a path fault occurring (S1201). The path management section 124a may include path fault or recovery

information in the LU management table update request so that the blocks setup management section 124b can make the above determination. Or, using the path ID notified with the LU management table update request, the blocks setup management

5    section 124b may reference the path management table 2100, derive the status of the path from the table, and make the above determination.

As a result, if the update request is due to a path fault occurring, the blocks setup management section 124b

10   decrements the number of paths by "1" in the LU management table 2200 (S1202). Then, the blocks setup management section 124b deletes the ID of the faulty path notified with the LU management table update request from the path IDs 2204 field (S1203). The blocks setup management section 124b

15   issues a request to update the block management table 2300 to the block management section 124c (S1204).

When receiving the LU management table update request issued by the path management section 124a, the blocks setup management section 124b determines whether the update

20   request is due to path recovery (S1205). As a result, if the update request is due to path recovery, the blocks setup management section 124b increments the number of paths by "1" in the L management table 2200 (S1206).

Then, the blocks setup management section 124b adds

25   the ID of the recovered path notified with the LU management table update request to the path IDs 2204 field (S1207). At this time, referring to the path management table 2100, the

blocks setup management section 124b gets information as to which logical unit that the ID of the recovered path is associated with.

Then, the blocks setup management section 124b issues 5 a request to update the block management table 2300 to the block management section 124c (S1208).

FIG. 13 is a flowchart of operation of the block management section 124c, according to an embodiment of the present invention.

10 The block management section 124c watches for a request to create the block management table from the blocks setup management section 124b. When receiving the request to create the block management table from the blocks setup management section (S1301), the block management section 15 124c references the LU management table 2200 and gets the number of active paths set for each logical unit and the capacity (size) of each logical unit (S1302, S1303). Then, the block management section 124c determines whether the blocks assignment table 2400 exists (S1304).

20 If the blocks assignment table 2400 exists, the block management section 124c judges the variable blocking mode on, determines the number of blocks into which a logical unit is divided, according to the contents of the blocks assignment table 2400, and assigns block IDs to the blocks 25 (S1305). For example, blocks IDs can be assigned to the blocks from "0" in ascending order of start address specified in the blocks assignment table 2400.

Then, the block management section 124c determines the start address of each block, referring to the blocks assignment table 2400 (S1306). Therefore, the logical unit is divided into blocks such that each block has capacity

5    determined by the range from the start address of the block to the start address of the next block. Then, the block management section 124c assigns the paths connected to each logical unit to the blocks and assigns the path IDs to the blocks (S1307).

10   Otherwise, if the blocks assignment table 2400 does not exist, the block management section 124c judges the even blocking mode on, divides each logical unit by the number of active paths into a plurality of blocks, and assigns block IDs to the blocks (S1308). The block management section 124c

15   divides the logical unit capacity by the number of blocks (the number of paths), thus calculating the size of each block (S1309). Thus, the logical unit is divided into blocks of even capacity.

Then, the block management section 124c determines the

20   start address of each block, based on the size of each block calculated in step S1309 (S1310). The block management section 124c assigns the paths connected to the logical unit to the blocks and assigns the path IDs to the blocks (S1311).

Accordingly, at the start of the path management

25   software, when receiving the request to create the block management table from the blocks setup management section 124b, the block management section 124c creates the block

management table 2300, referring to the LU management table 2200 (the blocks assignment table, as required). If the even blocking mode is one and the blocks assignment table 2400 is not created, the block management section 124c divides

5    each logical unit by the number of active paths into even size blocks and creates the block management table 2300. In the even blocking mode, all information necessary to create (or update) the block management table is stored in the LU management table.

10    The above-described procedures shown in FIGS. 10, 12, and 13 are sequentially performed at the start of the path management software and the path management table 2100, LU management table 2200, and block management table 2300 are created.

15    FIG. 14 is a flowchart of I/O processing, according to an embodiment of the present invention, and this processing is performed by the path management unit 120.

When the I/O request receiving section 121 receives an I/O request, the section to judge request sequential 125

20    judges whether the I/O request is sequential access I/O or random access I/O (S1401, S1402). To make this judgment, it is determined whether the I/O request is to get access to a location following a location specified in an I/O request of same type which was processed prior to the I/O request.

25    The I/O request judged sequential or random is queued into the I/O request queue 126.

As the result of judgment made by the section to judge

request sequential 125, when the I/O request is judged to be sequential access I/O, the I/O request allocation section 123 appoints one of the processing units 131 to 134 to handle the I/O request so that the I/O request is transmitted through

5  the same path that was used to transmit the last I/O request of same type and updates I/O processing unit appointment information (S1403). This I/O processing unit appointment information is a table in which the address to be accessed in the logical unit and the ID of the path used for the I/O

10  process are stored. If the start address of the I/O process requested by the I/O request succeeds (or falls within a given range from) the address specified in the table, the I/O request is judged sequential access and transmitted through the path whose ID is specified in this table (I/O processing

15  unit appointment information). If the I/O process is completed normally, the size of the data that has been read or written by this I/O process is added to the address specified in the table. Because the path used to transfer the I/O request is the same as the path specified in the table,

20  the path ID need not be updated. If the start address of the I/O process requested by the I/O request does not succeed (or falls outside the given range from) the address specified in the table, the I/O request is judged random access and a path is determined by reference to the block management

25  table 2300.

As the result of judgment made by the section to judge request sequential 125, when the I/O request is judged to

be random access I/O, the logical unit ID and the start
address to be accessed are extracted from the I/O request
(S1404). In some embodiment, it may also possible to skip
the judgment as to whether the I/O request is random or
5    sequential access I/O (that is, skip the steps S1401 to S1403)
and execute the step S1404 for all I/O requests and extract
the logical unit ID and the start address to be accessed from
the I/O requests.

By reference to the block management table 2300, the
10   logical unit to be accessed is determined from the logical
unit information and the block to be accessed is determined
from the start address. Then, the path assigned to the block
is determined from the path ID field in the block management
table 2300. Thus, the I/O request allocation section 123
15   appoints one of the processing units 131 to 134 to transmit
the I/O request through the path assigned to the block and
updates the I/O processing unit appointment information
(S1405).

Then, the I/O request is delivered to the appointed
20   I/O processing unit and the I/O request is processed by the
appointed I/O processing unit, one of 131 to 134 (S1406).
It is determined whether the I/O process is successful
(S1407). If the I/O process is successful, this procedure
terminates. If not, the section to detect path fault 123a
25   detects the path in which a fault has occurred and notifies
the path management section 124 of the path ID (S1408).

As described hereinbefore, in the embodiment of the

preset invention, a logical unit is divided into blocks, a cache memory is provided to prefetch and store both data in a location to be accessed by a data input/output request and data in locations following the location within the physical devices, requests for sequential access to a same block are transmitted through a same logical path. Consequently, the prefetch caching function can be used effectively even when the load balancing function is performed.

Even during sequential access, even if a random access request occurs occasionally, the random access request is allocated to a different path if it is for access to a different block. Therefore, the previously stored cache contents can be used to process an I/O request for access to a location following the location that was accessed by the last sequential access request. Particularly, if a plurality of applications gets access to one logical unit concurrently and even if requests for sequential access to different blocks are issued alternately, the cache contents can be used effectively.

To take full advantage of the above-mentioned prefetch caching, the size of blocks into which a logical unit is divided and the number of paths assigned to the blocks can be altered and performance of access to the logical unit can be enhanced. That is, by altering block size, I/O requests can be allocated to the paths evenly and access performance can be enhanced. By assigning a plurality of paths to a block, a load balancing function within the block can be achieved

effectively.   This load balancing function within the block is effective if random access to one block occurs more frequently.

When a logical path fault is detected or when the faulty logical path has recovered, reconfiguring a plurality of blocks into which a logical unit is divided is performed (for example, block size of the blocks and/or the number of paths assigned to the blocks are altered).   According to logical path status reconfiguring the blocks can be performed so that I/O processes are performed efficiently.